744777

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Harry Diamond Laboratories Washington, D.C. 20438	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP
3. REPORT TITLE EMPLIB: A SEQUENTIAL FILE PROGRAM LIBRARIAN	
4. DESCRIPTIVE NOTES (Type of report and inclusive dates)	
5. AUTHOR(S) (First name, middle initial, last name) William T. Wyatt, Jr.	
6. REPORT DATE April 1972	7a. TOTAL NO. OF PAGES 54 — 7b. NO. OF REFS 0
8a. CONTRACT OR GRANT NO. MIPR 0.00551 b. PROJECT NO. AMCMS Code: 5910.21.63388 c. HDL Proj No. E07E3 d.	9a. ORIGINATOR'S REPORT NUMBER(S) HDL-TR-1591 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
10. DISTRIBUTION STATEMENT Approved for public release; distribution unlimited	
11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Defense Nuclear Agency

13. ABSTRACT

EMPLIB, written for use on a CDC 6000 computer operating under Scope 3, is a librarian program whose function is to maintain an active library and a separate permanent archive of program UPDATE and object files on a sequential storage device such as a magnetic tape reel. The EMPLIB librarian can perform readout or alteration of the library or archive, and also certain file-positioning actions and program object file editing.

DD FORM 1 NOV 65 1473 REPLACES DD FORM 1473, 1 JAN 64, WHICH IS OBSOLETE FOR ARMY USE.

UNCLASSIFIED
Security Classification

A

UNCLASSIFIED
Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Computer Programs	8	3				
Librarians	8	3				
Information Retrieval	8	3				
Utility Programs	8	3				
FORTRAN	8	3				

B

UNCLASSIFIED
Security Classification

AD

MIPR 0.00551
AMCMS Code: 5910.21.63388
HDL Proj E07E3

HDL-TR-1591

EMPLIB: A SEQUENTIAL FILE PROGRAM LIBRARIAN

by
William T. Wyatt, Jr.

April 1972

This work was sponsored by the Defense Nuclear Agency under subtasks EA091 and EA094.

H•D•L

U.S. ARMY MATERIEL COMMAND
HARRY DIAMOND LABORATORIES
WASHINGTON, D.C. 20438

ABSTRACT

EMPLIB, written for use on a CDC 6000 computer operating under Scope 3, is a librarian program whose function is to maintain an active library and a separate permanent archive of program UPDATE and object files on a sequential storage device such as a magnetic tape reel. The EMPLIB librarian can perform readout or alteration of the library or archive, and also certain file-positioning actions and program object file editing.

Preceding page blank

CONTENTS

Preceding page blank

1. INTRODUCTION

EMPLIB is a p[illegible]written in CDC Fortran Extended and Compass for use on CDC 6000-series computers operating under Scope 3. It has been tested and run under Scopes 3.2 and 3.3, and requires about 54,200 words (octal) to load and execute. EMPLIB is a librarian program whose function is to maintain a library of frequently used program UPDATE files (called "source" files here) and program object files (called "binary" files here, i.e., the compiler or assembler object output). The term "file" is defined here as a string of data terminated by an EOF. The library is kept on a magnetic tape or other permanent sequential data storage device. EMPLIB also maintains an archive magnetic tape of program source or binary files to be saved indefinitely. The user may run the librarian program EMPLIB and cause it to perform certain library or archive functions by placing directive cards in the input card stream to be read by the librarian. These directive cards are processed sequentially, allowing library alteration, program file readout, user-assigned filenames for readin and readout functions, certain filename actions such as rewind, endfile, and skipfile, and archive additions or readout. The term "filename" is defined here as a logical file name (i.e., LGO, TAPE1, OLDPL, etc.).

2. THE LIBRARIAN

The librarian uses nine working filenames for various functions. All functions but one are assigned a one- or two-letter mnemonic and are associated by default with certain filenames which may be altered by the user during execution of the librarian. The file functions, mnemonics, default filenames, and purpose are listed below:

Function	Mnemonic	Filename	Purpose
card input	I	INPUT	Contains EMPLIB directives.
print output	O	OUTPUT	Contains printed output.
library	L	EMPLIB	Contains the program library.
archive	A	ARCHIV	Contains the program archive.
source input	SI	NEWPL	Source files read from NEWPL.
source output	SO	OLDPL	Source files written to OLDPL.
binary input	BI	LGO	Binary files read from LGO.
binary output	BO	XQT	Binary files written to XQT.
scratch	none	TAPE40	Scratch file for librarian.

All of the file functions, with the exception of the scratch function, may be assigned different filenames by use of the FILES directive described later. The filenames accessed by the librarian must all be odd-parity files as distinct from even-parity BCD files. The librarian can, of course, access an odd parity file onto which a BCD file has been copied. The terms file and binary file as used here both refer to files with odd parity. The difference between the two types of files is one of name only, and is conventionalized so that program UPDATE files are designated as source files and program object files are designated as binary files. The directives SELECT and REFUSE work properly only with binary files that are, in fact, program object files. Otherwise, any data file may be treated as a program source or binary file and manipulated by the librarian. The first file on the library filename is intended to be the librarian program object file, where it may be easily copied off and executed. (If the library tape is executed directly, the system loader will unload the tape, preventing later access to the library.) The second file contains a table of the library contents. Subsequent files are source and binary files previously placed in the library. Each file is identified in the table of contents by a name identifier, a version identifier, a mode identifier (to distinguish whether it is a source file or a binary file), and date of entry into the library. The name and version identifiers must be from one to ten characters with no imbedded blanks or commas. The version identifier is optional and will be all blanks if not specified by the user. The archive filename contains two data files for each source or binary file kept on it. The first is a file containing the table of contents information about the source or binary file, and the second data file is the source or binary file itself. The end of the archive is denoted by a file containing just the one word "LAST."

3. LIBRARIAN DIRECTIVES

The various functions the librarian can perform will be illustrated through their use in the following examples. The completed output is listed in Appendix A. It is assumed in the first example that the librarian object file has been copied to the filename EMPLIB (so as to allow creation of the library) and that a magnetic tape has been assigned to the filename ARCHIV. Execution of the librarian causes the filename EMPLIB to be rewound if the first directive is not a FILES directive; thus the library filename must be changed immediately if it is not to be EMPLIB.

Directives are free-field, but must have a dollar sign in column one. Directives and identifiers must be separated by blanks, unless commas are required. The librarian will copy each directive card to the print file as encountered and then add a description of any action taken. On the print file, directives can be recognized by the single dollar sign, whereas statements originated by the librarian begin with "EMPLIB $$$."

3.1 CREATE

CREATE causes a library to be created on the filename attached to the library function. Physically, the first file is skipped on the library filename and a table of contents file is written which records the first file as "EMPLIB" and the second file as "TOC." The library must be created (establishing a table of contents) before any library additions can be performed. In fact, a table of contents is required by all but the following directives: CREATE, CREATEARCH, FILES, SKIP, SKIPB, HISTORY, ENDFILE, REWIND, FIND, AND FINDB. The directives SELECT and REFUSE may or may not require a table of contents. $CREATE

3.2 CREATEARCH

CREATEARCH causes an archive to be established on the filename attached to the archive function. Physically, the hollerith word "LAST" is written on the archive. The archive must be established before any archive additions can be performed. The archive is rewound before and after the creation. $CREATEARCH

3.3 End of librarian input

The sequence of directives is terminated by a 7-8-9 card. If the last operation on the source output or binary output filename was a write-end-of-record, the filename is EOF'd and backspaced before execution is ended. All the following directives may be given in the same or any subsequent execution of the librarian once the library has been created. The file name/versions specified must be in the library when the directive is processed by the librarian, except for new name/versions in ADD (ADDB) and RENAME (RENAMEB), and except for the archive directive FIND (FINDB). File name/versions appearing with the FIND (FINDB) directive must already be on the archive when the directive is processed.

3.4 ADD and ADDB

ADD (ADDB) causes the source (binary) file on the source (binary) input filename to be added to the library, and assigns it to the name and version specified. The source (binary) input filename is rewound before reading is begun, unless suppressed by a NOREWIND directive (discussed later). $ADD PROG VERS

3.5 TOC

TOC causes a table of contents of the library to be printed. $TOC

3.6 FILES

FILES causes the file functions whose mnemonics are specified on the directive card to be reassigned different filenames. A reassignment consists of the mnemonic, one or more blanks, and the new filename, in that order. Multiple reassignments must be separated by commas. Old filenames whose last operation was a write-end-of-record are EOF'd and back-spaced before being detached from a file function when the reassignment is made. This directive may be issued even if the library has not been created. $FILES SI OLD, BI AGO, SO NEW

3.7 SKIP and SKIPB

SKIP (SKIPB) causes the number of files specified to be skipped in a forward direction on the source (binary) input filename. Up to 999 files may be skipped with one directive. If the number of files to be skipped is not specified, one file is skipped. $SKIPB 2

3.8 NOREWIND

NOREWIND suppresses the automatic rewind of the source (binary) input filename for the next (and only the next) ADD (ADDB) or CHANGE (CHANGEB) directive encountered. $NOREWIND

3.9 CHANGE and CHANGEB

CHANGE (CHANGEB) causes the source (binary) file name/version specified to be replaced on the library by the next file encountered on the source (binary) input filename. The filename is automatically rewound before reading unless suppressed, as in this example, by a NOREWIND directive. The present data is placed in the library table of contents for the file changed. The file changed must already be in the library. $CHANGE PROG VERS

3.10 RENAME and RENAMEB

RENAME (RENAMEB) causes the first source (binary) file name/version given on the card to be renamed in the table of contents file with the second source (binary) file name/version given on the card. The first and second file name/version must be separated by a comma. $RENAME PROG VERS, PROGA NEWNAME

3.11 DROP and DROPB

DROP (DROPB) causes the source (binary) file name/version to be removed from the library and its entry in the table of contents file to be deleted. The first file on the library (the EMPLIB binary file) will never be dropped, since this will cause the library to be scrambled. $DROP NEWPROG

3.12 KEEP and KEEPB

KEEP (KEEPB) causes the source (binary) file name/version specified to be added to the archive. The specified file name/version must already be in the library. Once added to the archive, a file cannot be removed from the archive by the librarian. $KEEPB PROG VERS

3.13 HISTORY

HISTORY causes the contents of the archive to be scanned and a list of the file name/versions encountered to be printed. This directive may be processed by the librarian even if the library has not been created; only the archive need exist. $HISTORY

3.14 RUN

RUN causes the first binary file on the library with the specified name to be copied to the binary output filename irrespective of the program version. Thus, only the program name need be specified. The terminating EOF is not copied, so further material may be copied to the binary filename to complete the desired load module. $RUN PROG

3.15 COPY and COPYB

COPY (COPYB) causes the source (binary) file name/version specified on the library to be copied to the source (binary) output filename. The terminating EOF is not copied, just as for the RUN directive. $COPY PROGA NEWNAME

3.16 SELECT

SELECT causes the specified binary file name/version on the library to be scanned for the named object programs or subprograms, which are copied as encountered to the binary output filename with no terminating EOF. On the directive card the binary file name/version must be the first identifiers after the SELECT word, followed by a comma, and then followed by the program or subprogram names separated by commas. If the file name/version is omitted so that the next non-blank character after the directive is a comma, the next file on the binary input filename will be scanned for the named programs and subprograms; this action does not require a table of contents. If the last non-blank character on the card is a comma, continuation cards will be read until the final non-blank card character is not a comma. Continuation cards must not contain a dollar sign in column one, and must contain information in columns one through 79 only. Up to 100 program or subprogram names may be specified in the directive. A list of all object routines encountered and their selection or refusal is printed. The largest object routine that can be processed by SELECT or REFUSE must be less than 6000 words long. A statement of the maximum size processed is printed at end of execution. $SELECT PROG VERS, ISO, SPLITR, SPLITC.

$SELECT, ISO, SPLITR, SPLITC.

3.17 REFUSE

REFUSE causes the same action as SELECT, except that specified object program and subprogram names are not copied to the binary output filename and all others encountered are copied. Empty records are not copied. Up to 100 names may be specified for refusal. If the file name/version is omitted, the binary input filename will be processed instead. $REFUSE PROG VERS, ISO, SPLITR, SPLITC

3.18 ENDFILE

ENDFILE causes the file function whose mnemonic is specified to have an EOF written on the filename assigned to the file function. Only one file function may be specified on the directive card and only the file functions BO and SO may be endfiled with this directive. $ENDFILE BO

3.19 REWIND

REWIND causes the file function whose mnemonic is specified to have its assigned filename rewound. If information had been written to the filename, end-of-information terminators are written to the filename before it is rewound. The file functions I,O, and L cannot be rewound with this directive. $REWIND SO

3.20 FIND and FINDB

FIND (FINDB) causes the archive to be searched for the source (binary) file name/version specified, which is then copied to the source (binary) output filename. No EOF is written, just as for the COPY (COPYB) directive. The directives FIND and FINDB may be processed by the librarian even if the library has not been created; only the archive is required to exist. $FINDB PROG VERS

3.21 REPLACE and REPLACEB

REPLACE (REPLACEB) causes the source (binary) file name/version specified to be replaced on the library on the next file encountered on the source (binary) input filename, and given a new name/version label. This combines the functions of CHANGE (CHANGEB) and REPLACE (REPLACEB). The directive format is the same as for the RENAME (RENAMEB) directive. The source (binary) input filename is rewound before reading is begun, unless suppressed by a NOREWIND directive.

4. LIBRARIAN ERROR MESSAGES

When the librarian detects an error involving the directive card being processed, a message describing the nature of the error is printed and the rest of the librarian card input file is copied to the print output file, after which execution is terminated by a call to the nonexistent subroutine ABORT which causes a mode one (address-out-of-range) error termination.

Terminators are assured to be on any source or binary output filename if the filename has been written on, just as for a normal termination.

If another kind of error is detected, an informative message is printed and execution is terminated immediately with a CALL ABORT. Terminators are not assured for filenames assigned to output functions at the time the error was detected.

4.1 Directive format errors

The following errors use the ABORT termination after checking file terminators:

1. Missing or misplaced dollar sign on directive card. The dollar sign must be in column one.

2. Improper directive. A directive cannot be found on the card.

3. Unrecognizable directive. The specified directive is not familiar to the librarian.

4. Directive requires a table of contents. The specified directive requires a created library when none exists.

5. Missing program filename. A program file name cannot be found on the card when one is required.

6. Program file name too long. The specified program file name is longer than 10 characters.

7. Program file version too long. The specified program file version is longer than 10 characters.

8. Program file name/version not in table of contents. The specified name/version is not on the library and the directive cannot be executed.

9. Adding file already in table of contents. The specified name/version/mode is already in the library; a unique name/version/mode must be specified.

10. Missing comma. A needed comma is missing between the old name/version and the new name/version on a RENAME, RENAMEB, REPLACE, or REPLACEB directive.

11. Word is too long. A word is longer than 10 characters on a FILES directive card. In fact, SCOPE can handle filenames only up to seven characters long, so care should be taken not to use 8, 9, or 10 character filenames.

12. Unrecognized file type. The file function type specified is not recognized.

13. More than 100 record names. Too many program and subprogram names are listed in a SELECT or REFUSE directive.

14. Illegal file type. A file function type cannot be found on the directive card.

15. Illegal directive for the file type. The directive is not allowed for the file function type specified.

16. Illegal number. Unrecognizable number on a SKIP or SKIPB card; 999 is the maximum allowed.

17. Program file name/version not on archive. The name/version specified by a FIND or FINDB directive is not in the archive.

4.2 Other errors

The following errors cause an immediate CALL ABORT termination:

1. KEEP read parity error. A read parity error occurred while reading the library for a KEEP or KEEPB directive.

2. KEEP write parity error. A write parity error occurred while writing to the archive for a KEEP or KEEPB directive. The former contents of the archive are intact, but an end-of-archive record no longer exists.

3. FIND read error. A read parity error occurred while reading the archive for a FIND or FINDB directive.

4. HISTORY read error. A read parity error occurred while reading the archive for a HISTORY directive.

5. GETTOC parity error. A read parity error occurred while the librarian was trying to read the table of contents file.

6. Empty file. The filename specified as the location of a program file was empty.

7. CPYFIL read parity error. A read parity error occurred while the librarian was skipping a file.

8. I/O error in CPYBUF. An I/O error occurred while the librarian was copying a file.

9. End-of-information encountered. An EOI was encountered while trying to copy a file; i.e., the filename was short-terminated.

10. TOC write parity error in NEWFIL. A write parity error occurred while the librarian was writing the table-of-contents file to the library for a library alteration directive.

11. Read error in CPYREC. A read parity error occurred while the librarian was reading or binary input filename during processing of a SELECT or REFUSE directive.

12. Write error in CPYREC. A write parity error occurred while the librarian was copying a program or subprogram record to the binary output filename during processing of a SELECT or REFUSE directive.

5. USER HINTS

The following information will be useful to the EMPLIB user.

5.1 List of directives

PROG and PROGA are program file names, and VERS and VERSA are program file versions in the following examples. Items enclosed in parenthesis are optional. An asterisk denotes the file is rewound before reading unless suppressed by a NOREWIND directive.

Input	Output	Directive
L	L	$CREATE
	A	$CREATEARCH
SI*	L	$ADD PROG (VERS)
BI*	L	$ADDB PROG (VERS)
L		$TOC
		$FILES BI ABC, A PQR7826
SI		$SKIP (5)
BI		$SKIPB (999)
		$NOREWIND
SI*	L	$CHANGE PROG (VERS)
BI*	L	$CHANGEB PROG (VERS)
L	L	$RENAME PROG (VERS), PROGA (VERSA)
L	L	$RENAMEB PROG (VERS), PROGA (VERSA)
L	L	$DROP PROG (VERS)
L	L	$DROPB PROG (VERS)
L	A	$KEEP PROG (VERS)
L	A	$KEEPB PROG (VERS)
A		$HISTORY
L	BO	$RUN PROG
L	SO	$COPY PROG (VERS)
L	BO	$COPYB PROG (VERS)
L	BO	$SELECT PROG (VERS), SUBA, SUBB, SUBC
BI	BO	$SELECT, SUBA, SUBB, SUBC
L	BO	$REFUSE PROG (VERS), SUBA, SUBB, SUBC
BI	BO	$REFUSE, SUBA, SUBB, SUBC
	BO or SO	$ENDFILE BO
All but I,O,L		$REWIND SI
A	SO	$FIND PROG (VERS)
A	BO	$FINDB PROG (VERS)
SI*	L	$REPLACE PROG (VERS), PROGA (VERSA)
SI*	L	$REPLACEB PROG (VERS), PROGA (VERSA)

All directives except CREATE which use the library (L) as input or output require a created library. All directives except CREATEARCH which use the archive (A) as input or output require a created archive.

5.2 File Actions

The librarian checks the first directive encountered and, if it is not FILES directive, rewinds the library (which has the filename EMPLIB) and looks to see if a table of contents exists. If it is a FILES directive, rewinding the library file is deferred to just prior to processing the next directive.

All directives which use the library as output cause the entire library to be copied to the scratch filename TAPE40 and recopied in its modified form back to the library filename. If the library is of substantial length and if more than one or two directives of this kind are to be processed, much PP time will be saved if the library tape is copied to a disk filename before librarian execution and then recopied from the disk filename back to the library tape after librarian execution. The library filename must be the disk filename, of course. This also helps protect the library tape from write parity errors.

A good practice is periodically to copy the entire library and the entire archive to a backup library tape and a backup archive tape, to avoid loss of program files if the first-line copies are impaired by permanent parity errors.

If the library is of short length, it may be practical to have the library reside on a permanent disk file instead of on a magnetic tape. The archive will generally be too large for this, however.

5.3 Examples of usage

Although it would not be possible to illustrate all the possible uses of the EMPLIB librarian, a few examples will be useful to convey the flexibility and simplicity of the program. The examples are for a Scope 3.3 system. All TOC directives are optional, but are recommended.

1. Update, compilation of changes, and execution.

```
JOB, CM54000, TP1
REQUEST, EMPLIB. (540/NORING)
COPYBF (EMPLIB, LIB, 1)
LIB.
RETURN (EMPLIB)
UPDATE (P)
FTN (I=COMPLE)
REWIND (XQT)
COPYBF (XQT, LGO, 1)
LGO.
7-8-9
$TOC
$COPY NEPHI CORRQ
$REFUSE NEPHI CORRQ, PHOTON, GROUND
7-8-9
(Update input with changes for subroutines PHOTON and GROUND.)
7-8-9
(Input data.)
6-7-8-9
```

This could also be accomplished by the following cards between the FTN card and UPDATE input cards:

```
LGO.
7-8-9
$TOC
$FILES BO LGO
$COPY NEPHI CORRQ
$REFUSE NEPHI CORRQ, PHOTON, GROUND
7-8-9
```

2. Update, compilation of changes, and alteration of library.

```
JOB, CM54000, TP1.
REQUEST, EMPLIB (540/RING)
COPYBF (EMPLIB, LIB, 1)
LIB.
UPDATE (P, N, W) (W makes new UPDATE library sequential.)
FTN (I = COMPILE)
LIB.
UNLOAD (EMPLIB)
7-8-9
$TOC
$COPY NEPHI CORRQ
$FILES BO LGO
$REFUSE NEPHI CORRQ, PHOTON, GROUND
7-8-9
(Update input with changes for subroutines PHOTON and GROUND.)
7-8-9
$DROP NEPHI CORRQ
$DROPB NEPHI CORRQ
$ADD NEPHI CORRR
$ADDB NEPHI CORRR
$TOC
6-7-8-9
```

More efficient use of the greater speed of disk files would be made by using the following control cards in the previous example:

```
JOB CM54000, TP1.
REQUEST, ZAP. (540/RING)
COPYBF (ZAP, EMPLIB, 100) (less than 100 files on ZAP)
EMPLIB.
UPDATE (P,N,W)
FTN (I = COMPILE)
EMPLIB.
REWIND (EMPLIB, ZAP)
COPYBF (EMPLIB, ZAP, LGO)
UNLOAD (ZAP)
7-8-9
```

3. Execution of one program.

```
JOB, CM54000, TP1.
REQUEST, EMPLIB (540/NORING.)
COPYBF (EMPLIB, LIB, 1)
LIB.
RETURN (EMPLIB)
RFL,100000.
REDUCE.
XQT.
7-8-9
$RUN NEPHI
7-8-9
```

```
(Input data for NEPHI.)
6-7-8-9
```

4. Execution of several programs.

```
JOB, CM54000, TP1.
REQUEST, EMPLIB (540/NORING.)
COPYBF (EMPLIB, LIB, 1)
LIB.
RETURN (EMPLIB)
XQT.
NEXT.
LAST.
7-8-9
$TOC
$COPYB PROG FIRST
$FILES BO NEXT
$RUN PROGSEC
$FILES BO LAST
$RUN PROGFIN
7-8-9
(Data for PROG/FIRST.)
7-8-9
(Data for PROGSEC.)
7-8-9
(Data for PROGFIN.)
6-7-8-9
```

Appendix A. SAMPLE OUTPUT

```
EMPLIB $$$ THE DATE IS  10/29/71  AND THE WORK FILES ARE
        BINARY OUTPUT = XQT             BINARY INPUT  = LGO
        SOURCE OUYPUT = OLDPL           SOURCE INPUT  = NEWPL
        EMP LIBRARY   = EMPLIB          ARCHIVE KEEP  = ARCHIV
        EMPLIB OUTPUT = OUTPUT          EMPLIB INPUT  = INPUT

$CREATE
EMPLIB $$$ TOC MISSING ON EMPLIB .
EMPLIB $$$ CREATED EMPLIB ON FILE NAMED EMPLIB .

$CREATEARCH
EMPLIB $$$ CREATED ARCHIVE ON FILE NAMED ARCHIV .

EMPLIB $$$ FINISHED $$$
EMPLIB $$$ THF DATE IS  10/29/71  AND THE WORK FILES ARE
           BINARY OUTPUT = XQT              BINARY INPUT  = LGO
           SOURCE OUTPUT = OLDPL            SOURCE INPUT  = NEWPL
           EMP LIBRARY   = EMPLIB           ARCHIVE KEEP  = ARCHIV
           EMPLIB OUTPUT = OUTPUT           EMPLIB INPUT  = INPUT

$ADD PROG VERS
EMPLIB $$$ ADDED  3TH FILE (SOURCE PROG      VERS       10/29/71 ) TO EMPLIB  FROM NEWPL   FILE.

$ADDB PROG VERS
EMPLIB $$$ ADDED  4TH FILE (BINARY PROG      VERS       10/29/71 ) TO EMPLIB  FROM LGO     FILE.

$TCC
EMPLIB $$$ TABLE OF CONTENTS OF EMPLIB
                      1          EMPLIB                                  10/29/71          BINARY
                                 TOC                                     10/29/71     4 FILES ON LIBRARY
                      3          PROG            VERS                    10/29/71          SOURCE
                      4          PROG            VERS                    10/29/71          BINARY

$FILES SI OLD, BI AGO, SO NEW
EMPLIB $$$ MADE SOURCE INPUT  FILE OLD     INSTEAD OF NEWPL  .
EMPLIB $$$ MADE BINARY INPUT  FILE AGO     INSTEAD OF LGO    .
EMPLIB $$$ MADE SOURCE OUTPUT FILE NEW    .

$SKIPB 2
EMPLIB $$$ SKIPPED   2 FILES ON AGO    .

$NOREWIND

$CHANGEB PROG VERS
EMPLIB $$$ CHANGED  4TH FILE (WAS BINARY PROG      VERS       10/29/71 , IS NOW BINARY PROG      VERS       10/29/71 )
                  ON EMPLIB  USING CONTENTS OF AGO     FILE.

$TOC
EMPLIB $$$ TABLE OF CONTENTS OF EMPLIB
                      1          EMPLIB                                  10/29/71          BINARY
                                 TOC                                     10/29/71     4 FILES ON LIBRARY
                      3          PROG            VERS                    10/29/71          SOURCE
                      4          PROG            VERS                    10/29/71          BINARY

$RENAME PROG VERS, PROGA NEWNAME
EMPLIB $$$ RENAMED  3TH FILE (WAS SOURCE PROG      VERS       10/29/71 , IS NOW SOURCE PROGA     NEWNAME    10/29/71 ).

$ ADD   NEWPROG
EMPLIB $$$ ADDED  5TH FILE (SOURCE NEWPROG              10/29/71 ) TO EMPLIB  FROM OLD     FILE.

$TOC
EMPLIB $$$ TABLE OF CONTENTS OF EMPLIB
                      1          EMPLIB                                  10/29/71          BINARY
                                 TOC                                     10/29/71     5 FILES ON LIBRARY
                      3          PROGA           NEWNAME                 10/29/71          SOURCE
                      4          PROG            VERS                    10/29/71          BINARY
                      5          NEWPROG                                 10/29/71          SOURCE

$DROP NEWPROG
EMPLIB $$$ DROPPED  5TH FILE (SOURCE NEWPROG              10/29/71 ) FROM EMPLIB .

$   TOC
EMPLIB $$$ TABLE OF CONTENTS OF EMPLIB
                      1          EMPLIB                                  10/29/71          BINARY
                                 TOC                                     10/29/71     4 FILES ON LIBRARY
                      3          PROGA           NEWNAMF                 10/29/71          SOURCE
```

```
                   4          PROG               VERS                 10/29/71          BINARY

$ KEEPB  PROG   VERS
EMPLIB $$$ KEPT   4TH FILE (BINARY PROG      VERS       10/29/71 ) FROM EMPLIB  ON ARCHIV  FILE.
                   1 SOURCE AND BINARY FILES NOW KEPT ON ARCHIV  FILE.

$KEEP PROGA  NEWNAME
EMPLIB $$$ KEPT   3TH FILE (SOURCE PROGA     NEWNAME    10/29/71 ) FROM EMPLIB  ON ARCHIV  FILE.
                   2 SOURCE AND BINARY FILES NOW KEPT ON ARCHIV  FILE.

$HISTORY
EMPLIB $$$ HISTORY OF ARCHIV
                    KEEP NO.   1          PROG      VERS       10/29/71 BINARY
                    KEEP NO.   2          PROGA     NEWNAME    10/29/71 SOURCE

$RUN PROG
EMPLIB $$$ COPIED  4TH FILE (BINARY PROG      VERS       10/29/71 ) FROM EMPLIB  TO XOT     FILE.

$COPY PROGA NEWNAME
EMPLIB $$$ COPIED  3TH FILE (SOURCE PROGA     NEWNAME    10/29/71 ) FROM EMPLIB  TO NEW     FILE.

$SELECT PROG VERS,ISO,SPLITR, SPLITC
EMPLIB $$$ COPYING THE FOLLOWING BINARY RECORDS ONTO XOT     FROM THE  4TH FILE (BINARY PROG      VERS       10/29/71 ) ON EMPLIB .
                                              SELECTED       REFUSED

                                              ISO
                                                             PLOTR
                                                             LLL
                                              SPLITR
                                              SPLITC
                                                             QUBFIT
                                                             FINDER
                                                             LODVAL
                                                             REDETC
                                                             RITETC
                                                             RSTRAN
                                                             DMPRAN
                                                             RITRAN
                                                             REDRAN
                                                             BLANKS
                                                             EXPON
                                                             LINES
                                                   -END OF COPY-

$REFUSE PROG VERS, ISO, SPLITR, SPLITC
EMPLIB $$$ COPYING THE FOLLOWING BINARY RECORDS ONTO XOT     FROM THE  4TH FILE (BINARY PROG      VERS       10/29/71 ) ON EMPLIB .
                                              SELECTED       REFUSED

                                                             ISO
                                              PLOTR
                                              LLL
                                                             SPLITR
                                                             SPLITC
                                              QUBFIT
                                              FINDER
                                              LODVAL
                                              REDETC
                                              RITETC
                                              RSTRAN
                                              DMPRAN
                                              RITRAN
                                              REDRAN
                                              BLANKS
                                              EXPON
```

```
                                            LINES
                                                 -END OF COPY-

$ENDFILE BO
EMPLIB $$$ ENDFILED BO FILE NAMED XOT    .

$REWIND SO
EMPLIB $$$ REWOUND  SO FILE NAMED NEW    .

$FILES BO BGO
EMPLIB $$$ MADE BINARY OUTPUT FILE BGO     INSTEAD OF XOT    .

$FINDB PROG VERS
EMPLIB $$$  1TH FILE FOUND (BINARY PROG      VERS       10/29/71 ) ON ARCHIV .
EMPLIB $$$ COPIED FILE FOUND TO BGO    .

$FILES BO CGO, BI BGO
EMPLIB $$$ MADE BINARY OUTPUT FILE CGO     INSTEAD OF BGO    .
EMPLIB $$$ MADE BINARY INPUT  FILE BGO     INSTEAD OF AGO    .

$REWIND BI
EMPLIB $$$ REWOUND  BI FILE NAMED BGO    .

$SELECT  ,ISO,SPLITR ,SPLITC
EMPLIB $$$ COPYING THE FOLLOWING BINARY RECORDS ONTO CGO     FROM BGO    .
                                            SELECTED       REFUSED

                                            ISO
                                                           PLOTR
                                                           LLL
                                            SPLITR
                                            SPLITC
                                                           QUBFIT
                                                           FINDER
                                                           LOOVAL
                                                           REDETC
                                                           RITETC
                                                           RSTRAN
                                                           OMPRAN
                                                           RITRAN
                                                           REDRAN
                                                           BLANKS
                                                           EXPON
                                                           LINES
                                                 -END OF COPY-

EMPLIB $$$ MAXIMUM RECORD LENGTH PROCESSED FOR SELECT-REFUSE WAS 2838.  5999 IS MAXIMUM ALLOWED.

EMPLIB $$$ FINISHED $$$
```

Appendix B. PROGRAM LISTING

```
      PROGRAM EMPLIB(XQT,OLDPL,LGO,EMPLIB=40008,ARCHIV,INPUT=10008,OUTPU
     *T=10008,NEWPL,TAPE40,TAPE1=XQT,TAPE2=OLDPL,TAPE3=LGO,TAPE4=EMPLIB,
     *TAPE5=ARCHIV,TAPE6=INPUT,TAPE7=OUTPUT,TAPE8=NEWPL)
      COMMON /MXC/MX
      COMMON //LMAX,A(6000)
      COMMON /ARGS/NAME,IVERS,NAMOLD,IVOLD,NREC,NAMREC(100),JTOC,LASTF,
     *IARCH
      DIMENSION CARD(80),TOC(4,50),MODE(2)
      COMMON /FILES/FILNAM(9),FETS(9),X(1)
      DIMENSION CHAR(29)
      INTEGER TOC,OLDATE,DDATE,A,CHAR,DOL,CARD,FILNAM
      INTEGER X,FETS
      DATA CHAR/3HRUN,4HCOPY,5HCOPYB,6HCHANGE,7HCHANGEB,3HADD,4HADDB,4HD
     *ROP,5HDROPB,3HTOC,4HKEEP,5HKEEPB,6HCREATE,4HFIND,5HFINDB,7HHISTORY
     *,6HRENAME,7HRENAMEB,6HREFUSE,6HSELECT,5HFILES,6HREWIND,7HENDFILE,
     *8HNOREWIND,4HSKIP,5HSKIPB,10HCREATEARCH,7HREPLACE,8HREPLACEB/,NCHA
     *R/29/
      DATA DOL/1H$/,MODE/5HSOURCE,6HBINARY/,LASTH/4HLAST/
      LMAX=6000
      MX=0
      NOREW=0
      IFLAG=0
      JTOC=0
      LASTF=0
      IARCH=0
      IRS=0
      JCW=0
      IFIRST=0
      CALL FTMBIN(0,0)
      CALL DATE(DDATE)
      CALL GETFIL
      PRINT 1,DDATE,FILNAM(1),FILNAM(3),FILNAM(2),FILNAM(8),FILNAM(4),FI
     *LNAM(5),FILNAM(7),FILNAM(6)
    1 FORMAT(*1EMPLIB $$$ THE DATE IS *,A10,* AND THE WORK FILES ARE*/
     *10X,*BINARY OUTPUT = *,A7,10X,*BINARY INPUT  = *,A7,/10X,*SOURCE O
     *UTPUT = *,A7,10X,*SOURCE INPUT  = *,A7/10X,*EMP LIBRARY   = *,A7,1
     *0X,*ARCHIVE KEEP  = *,A7/10X,*EMPLIB OUTPUT = *,A7,10X,*EMPLIB INP
     *UT  = *,A7)
   10 CONTINUE
      READ 2,CARD
    2 FORMAT(80A1)
      IF(EOF(6).NE.0) GO TO 1000
      IF(CARD(1).EQ.DOL) GO TO 20
      PRINT 3
    3 FORMAT(*0EMPLIB $$$ INVALID CONTROL CARD FOLLOWS, JOB WILL BE ABOR
     *TED AFTER READING INPUT FILE.*)
      IFLAG=1
   20 PRINT 4,CARD
    4 FORMAT(1H0,80A1)
      IF(IFLAG.NE.0) GO TO 10
      IF(IFIRST.NE.0) GO TO 2500
      I=1
      CALL NEXTWD(CARD(2),I,J,K)
      IF(K.NE.0.OR.J.NE.CHAR(21)) GO TO 2500
      IFIRST=1
```

```
              GO TO 30
      2500    CONTINUE
              IF(JTOC.NE.0) GO TO 30
              CALL GETTOC(TOC,NFILES,JCR)
              IF(JCR.EQ.1.AND.JTOC.EQ.0) PRINT 17,FILNAM(4)
      17      FORMAT(* EMPLIB $$$ TOC MISSING ON *,A7,*.*)
              JTOC=1
              IF(JCR.EQ.1) JTOC=-1
              REWIND 4
              LASTF=0
      30      CALL ISIT(CHAR,CARD(2),NCHAR,JUMP,IFILE,TOC,ODATE,JCR,FILNAM)
              IF(JUMP.NE.0) GO TO 40
              PRINT 19
      19      FORMAT(* EMPLIB $$$ WILL ABORT AFTER READING INPUT FILE.*)
              IFLAG=1
              GO TO 10
      40      CONTINUE
              GO TO (60,60,60,70,70,80,80,90,90,100,110,110,150,160,160,210,230,
             *230,240,240,10,10,10,250,260,260,270,280,280),JUMP
      60      CONTINUE
              CALL POSFIL(4,LASTF,IFILE)
              K=1
              IF(JUMP.EQ.2) K=2
              CALL CPYFIL(4,K,0)
              LASTF=IFILE
              J=TOC(4,IFILE)
              PRINT 6,IFILE,MODE(J),(TOC(I,IFILE),I=1,3),FILNAM(4),FILNAM(K)
      6       FORMAT(* EMPLIB $$$ COPIED *,I2,*TH FILE (*,A7,3A10,*) FROM *,A7,*
             * TO *,A7,* FILE.*)
              GO TO 10
      70      CONTINUE
      C  CHANGE AND CHANGED
              L=7HCHANGED
              NAMOLD=TOC(1,IFILE)
              IVOLD=TOC(2,IFILE)
      75      CONTINUE
              OLDATE=TOC(3,IFILE)
              TOC(3,IFILE)=ODATE
              REWIND 4
              REWIND 40
              CALL CPYFIL(4,40,NFILES)
              J=5
              IF(JUMP.EQ.5.OR.JUMP.EQ.29) J=3
              IF(NOREW.EQ.0) REWIND J
              NOREW=0
              CALL NEWFIL(NFILES,TOC,IFILE,J)
              LASTF=0
              K=TOC(4,IFILE)
              PRINT 7,L,IFILE,MODE(K),NAMOLD,IVOLD,OLDATE,MODE(K),(
             *TOC(I,IFILE),I=1,3),FILNAM(4),FILNAM(J)
      7       FORMAT(* EMPLIB $$$ *,A9,      I2,*TH FILE (WAS *,A7,3A10,*, IS NOW
             * *,A7,3A10,*)*/20X,*ON *,A7,* USING CONTENTS OF *,A7,* FILE.*)
              GO TO 10
      80      CONTINUE
      C  ADD AND ADDB
      J=8
      IF(JUMP.EQ.7) J=3
      REWIND 4
      REWIND 40
      IF(NOREW.EQ.0) REWIND J
      NOREW=0
      CALL CPYFIL(4,40,NFILES)
      NFILES=IFILE
      CALL NEWFIL(NFILES,TOC,NFILES,J)
      LASTF=0
      K=TOC(4,IFILE)
      PRINT 8,IFILE,MODE(K),(TOC(I,IFILE),I=1,3),FILNAM(4),FILNAM(J)
    8 FORMAT(* EMPLIB $$$ ADDED *,I2,*TH FILE (*,A7,3A10,*) TO *,A7,* FR
     *OM *,A7,* FILE.*)
      GO TO 10
   90 CONTINUE
C  DROP AND DROPB
      REWIND 4
      REWIND 40
      CALL CPYFIL(4,40,NFILES)
      J=TOC(4,IFILE)
      PRINT 9,IFILE,MODE(J),(TOC(I,IFILE),I=1,3),FILNAM(4)
      J=TOC(4,2)-1
      TOC(4,2)=J
      IF(J.EQ.IFILE-1) GO TO 96
      DO 95 I=IFILE,J
      DO 95 K=1,4
      TOC(K,I)=TOC(K,I+1)
   95 CONTINUE
   96 CONTINUE
      CALL NEWFIL(NFILES,TOC,IFILE,0)
      LASTF=0
      NFILES=NFILES-1
    9 FORMAT(* EMPLIB $$$ DROPPED *,I2,*TH FILE (*,A7,3A10,*) FROM *,A7,
     **.*)
      GO TO 10
  100 CONTINUE
C  TOC
      PRINT 11,FILNAM(4)
   11 FORMAT(* EMPLIB $$$ TABLE OF CONTENTS OF *,A7)
      DO 105 I=1,NFILES
      IF(I.EQ.2) GO TO 104
      K=TOC(4,I)
      PRINT 31,I,(TOC(J,I),J=1,3),MODE(K)
   31 FORMAT(I20,4(10X,A10))
      GO TO 105
  104 CONTINUE
      PRINT 32,(TOC(J,I),J=1,4)
   32 FORMAT(20X,3(10X,A10),I5,* FILES ON LIBRARY*)
  105 CONTINUE
      GO TO 10
C  KEEP AND KEEPB
  110 K=FETS(5)
      J=2*IARCH
      IF(IARCH.NE.0) GO TO 120
      I=0
      REWIND 5
  120 BUFFER IN(5,1) (A,A(2))
      I=I+1
      IF(UNIT(5)) 130,125,2000
  125 I=0
      J=J+1
      GO TO 120
  130 IF(LENGTH(5).NE.1) GO TO 120
      IF(A(1).NE.LASTH) GO TO 120
      IARCH=J/2
  140 CALL SKIPB(X(K),1)
      CALL POSFIL(4,LASTF,IFILE)
      BUFFER OUT(5,1) (TOC(1,IFILE),TOC(4,IFILE))
      IF(UNIT(5).GE.0) GO TO 2010
      ENDFILE 5
      CALL CPYFIL(4,5,1)
      LASTF=IFILE
      BUFFER OUT(5,1) (LASTH,LASTH)
      IF(UNIT(5).GE.0.0) GO TO 2010
      BACKSPACE 5
      IARCH=IARCH+1
      I=IARCH
      J=TOC(4,IFILE)
      PRINT 12,IFILE,MODE(J),(TOC(K,IFILE),K=1,3),FILNAM(4),FILNAM(5),I,
     *FILNAM(5)
   12 FORMAT(* EMPLIB $$$ KEPT *,I2,*TH FILE (*,A7,3A10,*) FROM *,A7,* O
     *N *,A7,* FILE.*/I20,* SOURCE AND BINARY FILES NOW KEPT ON *,A7,* F
     *ILE.*)
      GO TO 10
  150 CONTINUE
C  CREATE
      REWIND 4
      CALL CPYFIL(4,40,1)
      CALL NEWFIL(1,TOC,0,0)
      REWIND 4
      LASTF=0
      PRINT 16,FILNAM(4)
   16 FORMAT(* EMPLIB $$$ CREATED EMPLIB ON FILE NAMED *,A7,*.*)
      JTOC=0
      GO TO 10
  160 CONTINUE
C  FIND AND FINDB
      IARCH=0
      REWIND 5
      I=0
      M=1
      IF(JUMP.EQ.15) M=2
      J=1
  170 BUFFER IN(5,1) (A,A(4))
      I=I+1
      IF(UNIT(5)) 190,180,2020
  180 I=0
      J=J+1
      GO TO 170
      190   L=LENGTH(5)
            IF(A(1).NE.LASTH.OR.L.NE.1) GO TO 200
            PRINT 21,MODE(M),NAME,IVERS,FILNAM(5)
      21    FORMAT(* EMPLIB $$$ FILE SOUGHT (*,A7,2A10,*) IS NOT ON *,A7,*.*)
            IFLAG=1
            GO TO 10
      200   IF(L.NE.4) GO TO 170
            IF(M.NE.A(4).OR.NAME.NE.A(1).OR.IVERS.NE.A(2)) GO TO 170
            I=J/2+1
            PRINT 22,I,MODE(M),A(1),A(2),A(3),FILNAM(5)
      22    FORMAT(* EMPLIB $$$ *,I2,*TH FILE FOUND (*,A7,3A10,*) ON *,A7,*.*)
            BUFFER IN(5,1) (A,A)
            IF(UNIT(5).NE.0) GO TO 2020
            M=3-M
            CALL CPYFIL(5,M,0)
            PRINT 23,FILNAM(M)
      23    FORMAT(* EMPLIB $$$ COPIED FILE FOUND TO *,A7,*.*)
            IARCH=I
            GO TO 10
      210   CONTINUE
      C  HISTORY
            PRINT 25,FILNAM(5)
      25    FORMAT(* EMPLIB $$$ HISTORY OF *,A7)
            REWIND 5
            I=0
      220   I=I+1
            IARCH=I-1
            BUFFER IN(5,1) (A,A(4))
            IF(UNIT(5).GE.0) GO TO 2030
            IF(LENGTH(5).NE.1.OR.A(1).NE.LASTH) GO TO 225
            BACKSPACE 5
            GO TO 10
      225   CONTINUE
            K=A(4)
            PRINT 26,I,(A(J),J=1,3),MODE(K)
      26    FORMAT(21X,*KEEP NO.*,I4,10X,4A10)
            BUFFER IN(5,1) (A,A)
            IF(UNIT(5).NE.0) GO TO 2030
            CALL CPYFIL(5,0,1)
            GO TO 220
      230   CONTINUE
      C  RENAME AND RENAMEB
            REWIND 4
            REWIND 40
            CALL CPYFIL(4,40,NFILES)
            CALL NEWFIL(NFILES,TOC,0,0)
            LASTF=0
            K=TOC(4,IFILE)
            PRINT 28,IFILE,MODE(K),NAMOLD,IVOLD,TOC(3,IFILE),MODE(K),(TOC(I,IF
           *ILE),I=1,3)
      28    FORMAT(* EMPLIB $$$ RENAMED *,I2,*TH FILE (WAS *,A7,3A10,*, IS NOW
           * *,A7,3A10,*).*)
            GO TO 10
      240   CONTINUE
      C  REFUSE AND SELECT
      IRS=1
      IF(IFILE.EQ.0) GO TO 245
      CALL POSFIL(4,LASTF,IFILE)
      K=TOC(4,IFILE)
      PRINT 29,FILNAM(1),IFILE,MODE(K),(TOC(I,IFILE),I=1,3),FILNAM(4)
29    FORMAT(* EMPLIB $$$ COPYING THE FOLLOWING BINARY RECORDS ONTO *,A7
     *,* FROM THE *,I2,*TH FILE (*,A7,3A10,*) ON *,A7,*.*//45X,*SELECTED
     **,7X,*REFUSED*/)
      K=NREC
      IF(JUMP.EQ.20) K=-K
      CALL CPYREC(4,1,NAMREC,K)
      LASTF=IFILE
      GO TO 10
245   CONTINUE
      PRINT 36,FILNAM(1),FILNAM(3)
36    FORMAT(* EMPLIB $$$ COPYING THE FOLLOWING BINARY RECORDS ONTO *,A7
     *,* FROM *,A7*.*//45X,*SELECTED*,7X,*REFUSED*/)
      K=NREC
      IF(JUMP.EQ.20) K=-K
      CALL CPYREC(3,1,NAMREC,K)
      GO TO 10
250   CONTINUE
C  NOREWIND
      NOREW=1
      GO TO 10
260   CONTINUE
C  SKIP AND SKIPB
      I=8
      IF(JUMP.EQ.26) I=3
      CALL CPYFIL(I,0,IFILE)
      PRINT 33,IFILE,FILNAM(I)
33    FORMAT(* EMPLIB $$$ SKIPPED*,I4,* FILES ON *,A7,*.*)
      GO TO 10
270   CONTINUE
C  CREATEARCH
      REWIND 5
      IARCH=0
      A(1)=4HLAST
      BUFFER OUT(5,1) (A,A)
      IF(UNIT(5).GE.0.0) GO TO 2010
      REWIND 5
      PRINT 35,FILNAM(5)
35    FORMAT(* EMPLIB $$$ CREATED ARCHIVE ON FILE NAMED *,A7,*.*)
      GO TO 10
280   CONTINUE
C  REPLACE AND REPLACEB
      L=8HREPLACED
      GO TO 75
1000  CONTINUE
      M=LMAX-1
      IF(IRS.NE.0) PRINT 34,MX,M
34    FORMAT(*0EMPLIB $$$ MAXIMUM RECORD LENGTH PROCESSED FOR SELECT-REF
     *USE WAS*,I5,*.*,I6,* IS MAXIMUM ALLOWED.*)
      IF(IFLAG.EQ.0) PRINT 13
13    FORMAT(*0EMPLIB $$$ FINISHED $$$*)
```

```
              REWIND 4
              DO 1010 J=1,2
              I=FETS(J)
              IF((X(I).AND.508).NE.20) GO TO 1010
              ENDFILE J
              CALL SKIP8(X(I),0)
        1010  CONTINUE
              IF(IFLAG.EQ.0) STOP
              PRINT 18
        18    FORMAT(*DEMPLIB $$$ ABORTING $$$*)
              CALL ABORT
        2000  CONTINUE
              PRINT 14,I,J,FILNAM(5)
        14    FORMAT(* EMPLIB $$$ KEEP READ PARITY ERROR ON*,I5,*TH RECORD ON*,I
             *3,*TH FILE ON *,A7,*.*)
              CALL ABORT
        2010  CONTINUE
              PRINT 15,FILNAM(5)
        15    FORMAT(* EMPLIB $$$ KEEP WRITE PARITY ERROR ON *,A7,*.*)
              CALL ABORT
        2020  CONTINUE
              PRINT 24,FILNAM(5)
        24    FORMAT(* EMPLIB $$$ FIND READ ERROR ON *,A7,*.*)
              CALL ABORT
        2030  CONTINUE
              PRINT 27,FILNAM(5)
        27    FORMAT(* EMPLIB $$$ HISTORY READ ERROR ON *, A7,*.*)
              CALL ABORT
              END
```

```
      SUBROUTINE GETTOC(TOC,NFILES,JCR)
      DIMENSION TOC(4,50)
      REWIND 4
      JCR=0
      CALL CPYFIL(4,0,1)
      BUFFER IN(4,1) (TOC,TOC(4,50))
      IF(UNIT(4)) 10,20,100
   10 NFILES=LENGTH(4)/4
      RETURN
   20 NFILES=2
      JCR=1
      RETURN
  100 PRINT 1
    1 FORMAT(* EMPLIB $$$ GETTOC PARITY ERROR*)
      CALL ABORT
      END
      SUBROUTINE CPYFIL(IIN,IOUT,NF)
      COMMCN //LMAX,A(1)
      COMMCN /FILES/FILNAM(9),FETS(9),X(1)
      INTEGER FILNAM,FETS,X
      IF(IOUT.GT.0) GO TO 30
      DO 20 I=1,NF
10    BUFFER IN(IIN,1) (A,A)
      IF(UNIT(IIN)) 10,20,200
20    CONTINUE
      RETURN
30    CONTINUE
      LMX=512*(LMAX/512)
      JIN=MINO(9,IIN)
      JOUT=MINO(9,IOUT)
      J=FETS(JIN)
      K=FETS(JCUT)
      DO 40 I=1,NF
35    CONTINUE
      CALL CPYBUF(A,A(LMX+2),X(J),X(K),IER)
      IF(IER.NE.1) GC TO 36
      IF(I.NE.1) GO TO 400
      PRINT 3,FILNAM(IIN)
3     FORMAT(* EMPLIB $$$ *,A7,* INITIALLY PCSITICNEC AT END-OF-INFCRMAT
     *ION, EMPLIB ABCRTINC.*)
      CALL ABORT
36    CONTINUE
      IF(IER.NE.0) GC TO 300
      IF(NF.GT.0) ENCFILE IOUT
40    CONTINUE
      RETURN
200   CONTINUE
      PRINT 1,I,FILNAM(JIN)
1     FORMAT(* EMPLIB $$$ CPYFIL READ PARITY ERROR IN*,I3,*TH FILE (FRCM
     * START OF CCPY) ON *,A7,*.*)
      CALL ABCRT
300   CONTINUE
      PRINT 2,FILNAM(JIN),FILNAM(JOUT),I,IER
2     FORMAT(* EMPLIB $$$ I/O ERROR IN CPYBUF WHILE COPYING *,A8,*TC *,A
     *7,*, FILE NUMBER*,I3/20X,*ERROR CODE IN CCTAL IS *,O20)
      CALL ABCRT
400   CONTINUE
      PRINT 5,FILNAM(JIN),I,NF
5     FORMAT(* EMPLIB $$$ END-OF-INFORMATION ENCOUNTEREC COPYING*,I3,*TH
     * OF*,I3,* FILES (FROM START OF CCPY) ON *,A7,*.*)
      CALL ABCRT
      END
```

```
                    SUBROUTINE GETFIL
                    COMMON /FILES/FILNAM(9),FETS(9),X(1)
                    INTEGER FILNAM,FETS,X
                    DATA MASK/777777B/
                    L=LOCF(X)
                    DO 10 I=1,9
                    J=2-L+I
                    FILNAM(I)=X(J)
                    J=FILNAM(I).AND.MASK
                    FETS(I)=J-L+1
             10     CONTINUE
                    RETURN
                    END
```

```
          SUBROUTINE SWAPFIL(IUNIT,NAME)
          DIMENSION WORD1(8),WORD2(8)
          COMMON /FILES/FILNAM(9),FETS(9),X(1)
          INTEGER FILNAM,FETS,X
          DATA WORD1/10HBINARY OUT,10HSOURCE OUT,10HBINARY INP,10HEMP LIBRAR
         *,10HARCHIVE KE,10HEMPLIB INP,10HEMPLIB OUT,10HSOURCE INP/,WORD2/3H
         *PUT,3HPUT,3HUT ,3HY  ,3HEP ,3HUT ,3HPUT,3HUT /
          DATA MASKDT/77770000000000000000B/,MASK/777777B/,MASKCH/77B/,IB/1H
         * /
          I=FETS(IUNIT)
          II=X(I+1)
          KFLAG=0
          IF((II.AND.MASKDT).EQ.0.AND.(X(I+4).AND..NOT.MASK).EQ.0) GO TO 10
          KFLAG=1
          IN=X(I).AND.50B
C  CHECK TO SEE IF LAST OPERATION WAS AN FOR WRITE.
          IF(IN.NE.20) GO TO 5
          ENDFILE IUNIT
          CALL SKIP9(X(I),0)
    5     CONTINUE
          X(I+1)=X(I+1).AND..NOT.MASKDT
          X(I+4)=X(I+4).AND.MASK
   10     CONTINUE
          IN=NAME
          DO 20 J=4,10
          M=SHIFT(MASKCH,6*(J-1))
          IF((IN.AND.M).NE.(IB.AND.M)) GO TO 30
          IN=IN.AND..NOT.M
   20     CONTINUE
   30     CONTINUE
          X(I)=(IN.AND..NOT.MASK).OR.3
          I=FILNAM(IUNIT)
          FILNAM(IUNIT)=IN
          IF(KFLAG.GT.0) GO TO 40
          PRINT 1,WORD1(IUNIT),WORD2(IUNIT),FILNAM(IUNIT)
    1     FORMAT(* EMPLIB $$$ MADE *,A10,A3,* FILE *,A7,*.*)
          RETURN
   40     PRINT 2,WORD1(IUNIT),WORD2(IUNIT),FILNAM(IUNIT),I
    2     FORMAT(* EMPLIB $$$ MADE *,A10,A3,* FILE *,A7,* INSTEAD OF *,A7,*.
         **)
          RETURN
          END
```

```
                                     COMPASS - VER 2.     01/18/72  11.12.49.

          IDENT SKIPB
        ENTRY SKIPB
        EXT CPC
        VFD 30/5HSKIPB,30/1
SKIPB   BSS 1
        SX7 A0
        SA7 SAVA0
        SA2 A1+1
        SA3 X2
        SA1 X1
        NZ  X3,SKIPFIL
        RJ  CPC
        VFD 18/3,2/1,22/1,18/640B
        EQ  RET
SKIPFIL SA4 ARG
        LX3 18
        BX6 X3+X4
        SA6 ARGLOC
        RJ  CPC
ARGLOC  BSS 1
RET     SA5 SAVA0
        SA0 X5
        EQ  SKIPB
SAVA0   BSS 1
ARG     VFD 18/3,2/1,22/0,4/17B,14/640B
        END
STORAGE USED              26 STATEMENTS        7 SYMBOLS
6600 ASSEMBLY          0.121 SECONDS          16 REFERENCES
```

```
          IDENT CPYBUF
         ENTRY CPYBUF
         VFD 36/6HCPYBUF,24/5
CPYBUF   BSS 1
         SB7 1
         SB6 B7+B7
         SA2 A1+B7
         SA3 A1+B6                   X3 TO CONTAIN ADDRESS OF FILEIN
         SA4 A2+B6                   X4 SAME FOR FILEOUT
         SA5 A3+B6
         SX6 A0
         SA6 AZERO
         BX7 X5
         SA7 IER
         MX7 0
         SA7 X5
         SA7 FLAG
         BX6 X1
         SA6 BOUNDS
         BX7 X2
         SA7 A6+B7
         SA1 X3+B7
         SA2 A1+B7
         BX6 X1
         BX7 X2
         SA1 A2+B7
         SA2 A1+B7
         SA6 SAVE
         SA7 A6+B7
         BX6 X1
         BX7 X2
         SA6 A7+B7
         SA7 A6+B7
         SA1 X4+B7
         SA2 A1+B7
         BX6 X1
         BX7 X2
         SA1 A2+B7
         SA2 A1+B7
         SA6 A7+B7
         SA7 A6+B7
         BX6 X1
         BX7 X2
         SA6 A7+B7
         SA7 A6+B7
LOOP     SA1 BOUNDS
         SB2 X1
         SA2 A1+B7
         SB1 X2
         SA1 X3+B7
         MX0 42
         SX7 B2
         BX6 X1*X0
         BX6 X6+X7
         SA6 X3+B7
         SA7 A6+B7
         SA7 A7+B7
         SA1 A1+3
         SX2 B1
         BX6 X1*X0
         BX7 X6+X2
         SA7 A7+B7
         SX1 12B
         SA2 X3
         BX6 X0*X2
         BX7 X6+X1
         SA7 X3
         SA1 CIOWORD
         BX6 X1+X3
         SA6 B7
RECALL   SA5 B7
         NZ  X5,RECALL
         SA1 X3
         SX0 37000B
         BX2 X1*X0
         ZR  X2,OK
         SX0 3000B
         BX2 X1*X0
         NZ  X2,EOI
         SA5 IER
         BX6 X1
         SA6 X5
         EQ  RETURN
EOI      SX2 740033B
         MX0 42
         BX6 X0*X1
         BX6 X6+X2
         SA6 X3
         SA2 FLAG
         NZ  X2,RETURN
         SX6 B7
         SA2 IER
         SA6 X2
         EQ  RETURN
OK       SX0 77B
         BX7 X0*X1
         SX0 33B
         IX6 X7-X0
         ZR  X6,RETURN
         SA1  X3+B7
         SA2 X4+B7
         MX0 42
         BX6 -X0*X1
         BX7 X0*X2
         BX7 X6+X7
         SA7 A2
         SA1 A1+B7
         BX6 X1
         SA6 A7+B7
         SA2 A1+B7
         BX7 X2
         SA7 A6+B7
         SA1 A2+B7
         SA2 A7+B7
         BX6 -X0*X1
         BX7 X0*X2
         BX7 X6+X7
         SA7 A2
         SX6 B7
         SA7 FLAG
         SA1 X3
         SA2 X4
         SB3 X1
         SX6 B3+3
         BX7 X0*X2
         BX7 X6+X7
         SA7 X4
         SA1 CIOWORD
         BX6 X1+X4
         SA6 B7
RECALLA  SA5 B7
         NZ  X5,RECALLA
         SA1 X4
         SX0 370008
         BX2 X1*X0
         ZR  X2,OKA
         SA5 IER
         BX6 X1
         SA6 X5
         EQ  RETURN
OKA      SB1 -378
         SB2 X1+B1
         NE  B2,LOOP
RETURN   SA1 SAVE
         SA2 A1+B7
         MX0 42
         SA5 X3+B7
         BX6 -X0*X1
         BX5 X0*X5
         BX6 X6+X5
         BX7 X2
         SA1 A2+B7
         SA2 A1+B7
         SA5 X3+4
         SA6 X3+B7
         SA7 A6+B7
         BX6 X1
         BX7 -X0*X2
         BX5 X0*X5
         BX7 X7+X5
         SA6 A7+B7
         SA7 A6+B7
         SA1 A2+B7
         SA2 A1+B7
         SA5 X4+B7
         BX6 -X0*X1
         BX5 X0*X5
         BX6 X6+X5
         BX7 X2
         SA1 A2+B7
         SA2 A1+B7
         SA5 X4+4
            SA6 X4+B7
         SA7 A6+B7
         BX6 X1
         BX7 -X0*X2
         BX5 X0*X5
         BX7 X7+X5
         SA6 A7+B7
         SA7 A6+B7
         SA1 AZERO
         SA0 X1
         EQ  CPYBUF
IER      BSS 1
AZERO    BSS 1
BOUNDS   BSS 2
FLAG     BSS 1
SAVE     BSS 8
CIOWORD VFD 18/3HCIO,2/1,40/0
         END
STORAGE USED           190 STATEMENTS       14 SYMBOLS
6600 ASSEMBLY        0.536 SECONDS          42 REFERENCES
```

```
            SUBROUTINE NEWFIL(NFILES,TOC,IFILE,J)
            DIMENSION TOC(4,NFILES)
            REWIND 4
            REWIND 40
            L=MAX0(2,NFILES)
            K=L
            IF(IFILE.NE.0.AND.J.EQ.0) K=K-1
            DO 30 I=1,L
            IF(I.NE.2) GO TO 10
            BUFFER OUT(4,1) (TOC,TOC(4,K))
            IF(UNIT(4).GE.0.0) GO TO 100
            ENDFILE 4
            IF(NFILES.NE.1) CALL CPYFIL(40,0,1)
            GO TO 30
     10     IF(I.NE.IFILE) GO TO 20
            IF(NFILES.NE.IFILE) CALL CPYFIL(40,0,1)
            IF(J.NE.0) CALL CPYFIL(J,4,1)
            GO TO 30
     20     CALL CPYFIL(40,4,1)
     30     CONTINUE
            REWIND 4
            RETURN
    100     CONTINUE
            PRINT 1
      1     FORMAT(* EMPLIB $$$ TOC WRITE PARITY ERROR IN NEWFIL.*)
            CALL ABORT
            END
```

```
SUBROUTINE  POSFIL                                    CDC 6600 FTN V3.0-P270 OPT=1  0
                          SUBROUTINE POSFIL(N,LASTF,IFILE)
                          COMMON /FILES/FILNAM(9),FETS(9),X(1)
                          INTEGER FILNAM,FETS,X
                          IF(IFILE.GT.LASTF) GO TO 10
                          I=FETS(N)
                          CALL SKIPB(X(I),LASTF-IFILE+1)
                          GO TO 30
                   10     JFILES=IFILE-LASTF-1
                          IF(JFILES.EQ.0) GO TO 30
                          CALL CPYFIL(N,0,JFILES)
                   30     LASTF=IFILE-1
                          RETURN
                          END
```

```
      SUBROUTINE ISIT(CHAR,CARD,NCHAR,JUMP,IFILE,TOC,DDATE,JCR,FILNAM)
      COMMON /ARGS/NAME,IVERS,NAMOLD,IVOLD,NREC,NAMREC(100),JTOC,LASTF,
     *IARCH
      INTEGER FILTYP(8),NTYP(8)
      INTEGER FILNAM(1)
      DIMENSION CARD(79),TOC(4.1),CHAR(1)
      DIMENSION MODEH(2)
      DIMENSION NUMS(10)
      INTEGER CARD,TOC,DDATE,CHAR
      DATA FILTYP/2HBO,2HBI,2HSO,2HSI,1HL,1HA,1HO,1HI/,NTYP/1,3-2,8,4,5,
     *7,6/
      DATA IB/1H /,MASK/77B/
      DATA MODEH/6HSOURCE,6HBINARY/
      DATA NUMS/1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9/
      JUMP=0
      NAME=IB
      IST=1
      CALL NEXTWD(CARD,IST,NAME,JFLAG)
      IF(JFLAG.EQ.0) GO TO 10
      PRINT 1,CARD
    1 FORMAT(* EMPLIB $$$ IMPROPER DIRECTIVE ON CARD.**,79A1)
   10 CONTINUE
      DO 40 JUMP=1,NCHAR
      IF(CHAR(JUMP).EQ.NAME) GO TO 50
   40 CONTINUE
      PRINT 2,NAME,CARD
    2 FORMAT(* EMPLIB $$$ UNRECOGNIZABLE DIRECTIVE.**,A10,***,79A1)
      JUMP=0
      RETURN
   50 CONTINUE
      IF(JUMP.EQ.27) RETURN
      IF(JUMP.EQ.13) GO TO 300
      IF(JUMP.EQ.21) GO TO 500
      IF(JCR.EQ.0) GO TO 55
      IF(JUMP.GE.13.AND.JUMP.NE.17.AND.JUMP.NE.15) GO TO 55
      JUMP=0
      PRINT 7,NAME
    7 FORMAT(* EMPLIB $$$ DIRECTIVE REQUIRES TABLE OF CONTENTS, WHICH HA
     *S NOT BEEN CREATED.**,A10)
      RETURN
   55 IF(JUMP.EQ.10.OR.JUMP.EQ.16) RETURN
      IF(JUMP.EQ.24) RETURN
      IF(JUMP.EQ.25.OR.JUMP.EQ.26) GO TO 800
      IF(JUMP.EQ.22.OR.JUMP.EQ.23) GO TO 700
      NAME=IB
      CALL NEXTWD(CARD,IST,NAME,JFLAG)
      IF(JFLAG.NE.1.OR.(JUMP.NE.19.AND.JUMP.NE.20)) GO TO 56
      IFILE=0
      GO TO 600
   56 CONTINUE
      IF(JFLAG) 70,80,60
   60 CONTINUE
      PRINT 3,NAME,CARD
    3 FORMAT(* EMPLIB $$$ CANNOT FIND PROGRAM NAME ON CARD.**,79A1)
      JUMP=0
```

```
        RETURN
70      CONTINUE
        PRINT 4,NAME,CARD
4       FORMAT(* EMPLIB $$$ PROGRAM NAME TOO LONG.**,A10,***,79A1)
        JUMP=0
        RETURN
80      CONTINUE
        IVERS=10
        CALL NEXTWD(CARD,IST,IVERS,JFLAG)
        IF(JFLAG.GE.0) GO TO 100
        PRINT 5,IVERS,CARD
5       FORMAT(* EMPLIB $$$ VERSION NAME TOO LONG.**,A10,***,79A1)
        JUMP=0
        RETURN
100     CONTINUE
        IF(JFLAG.EQ.1) IST=IST-1
        IF(JUMP.EQ.14.OR.JUMP.EQ.15) RETURN
        N=TOC(4,2)
        MODE=1
        IF(JUMP.EQ.29) MODE=2
        IF(JUMP.EQ.3.OR.JUMP.EQ.5.OR.JUMP.EQ.9) MODE=2
        IF(JUMP.EQ.1.OR.JUMP.EQ.7.OR.JUMP.EQ.12.OR.(JUMP.GE.18.AND.JUMP.LE
       *.20)) MODE=2
        DO 160 IFILE=1,N
        IF(TOC(4,IFILE).EQ.MODE.AND.TOC(1,IFILE).EQ.NAME.AND.TOC(2,IFILE).
       *EQ.IVERS) GO TO 170
        IF(JUMP.NE.1.OR.TOC(4,IFILE).NE.MODE.OR.TOC(1,IFILE).NE.NAME) GO T
       *O 160
C  RUN
        IVERS=TOC(2,IFILE)
        RETURN
160     CONTINUE
        IF(JUMP.EQ.6.OR.JUMP.EQ.7) GO TO 180
        PRINT 6,NAME,IVERS
6       FORMAT(* EMPLIB $$$ *,2A10,* NOT IN TOC.*)
        JUMP=0
        RETURN
170     CONTINUE
        IF(JUMP.EQ.6.OR.JUMP.EQ.7) GO TO 185
C  RUN, COPY, COPYB, KEEP, AND KEEPB
        IF(JUMP.EQ.1.OR.JUMP.EQ.2.OR.JUMP.EQ.3.OR.JUMP.EQ.11.OR.JUMP.EQ.12
       *) RETURN
        IF(JUMP.NE.4.AND.JUMP.NE.5) GO TO 190
C  CHANGE AND CHANGEB
        RETURN
180     CONTINUE
C  ADD AND ADDB
        IFILE=TOC(4,2)+1
        TOC(4,2)=IFILE
        TOC(1,IFILE)=NAME
        TOC(2,IFILE)=IVERS
        TOC(3,IFILE)=DDATE
        TOC(4,IFILE)=JUMP-5
        RETURN
185     CONTINUE
```

```
      J=TOC(4,IFILE)
      PRINT 11,IFILE,MODEM(J),(TOC(I,IFILE),I=1,3)
11    FORMAT(* EMPLIB $$$ ADDING FILE ALREADY IN TOC IS NOT PERMITTED. *
     ** FILE IS *,I2,* FILE (*,A7,3A10,*).*)
      JUMP=0
      RETURN
190   IF(JUMP.NE.8.AND.JUMP.NE.9) GO TO 210
C  DROP AND DROPB
      IF(IFILE.GT.2) RETURN
      PRINT 8
8     FORMAT(* EMPLIB $$$ DROPPING LIBRARIAN OR TABLE OF CONTENTS IS NOT
     * PERMITTED.*)
      JUMP=0
      RETURN
210   IF(JUMP.NE.17.AND.JUMP.NE.18) GO TO 600
C  RENAME AND RENAMEB
      CALL NEXTWD(CARD,IST,NAME,JFLAG)
      IF(JFLAG.EQ.1) GO TO 220
215   PRINT 12,CARD
12    FORMAT(* EMPLIB $$$ MISSING COMMA.**,79A1)
      JUMP=0
      RETURN
220   NAME=IB
      CALL NEXTWD(CARD,IST,NAME,JFLAG)
      IF(JFLAG) 230,250,240
230   PRINT 4,NAME,CARD
      JUMP=0
      RETURN
240   PRINT 3,NAME,CARD
      JUMP=0
      RETURN
250   IVERS=IB
      CALL NEXTWD(CARD,IST,IVERS,JFLAG)
      IF(JFLAG.GE.0) GO TO 260
255   PRINT 5,IVERS,CARD
      JUMP=0
      RETURN
260   CONTINUE
      NAMOLD=TOC(1,IFILE)
      IVOLD=TOC(2,IFILE)
      TOC(1,IFILE)=NAME
      TOC(2,IFILE)=IVERS
      RETURN
300   CONTINUE
C  CREATE
      TOC(1,2)=3HTOC
      TOC(2,2)=IB
      TOC(3,2)=DDATE
      TOC(4,2)=2
      TOC(1,1)=6HEMPLIB
      TOC(2,1)=IB
      TOC(3,1)=DDATE
      TOC(4,1)=2
      JCR=0
      RETURN
```

```
500   CONTINUE
C  FILES
510   CALL NEXTWD(CARD,IST,NAME,JFLAG)
      IF(JFLAG.EQ.2) RETURN
      IF(JFLAG) 520,530,510
520   PRINT 13,NAME,CARD
13    FORMAT(* EMPLIB $$$ WORD IS TOO LONG.'*,A10,*'*79A1)
      JUMP=0
      RETURN
530   CONTINUE
      DO 540 I=1,8
      IF(NAME.EQ.FILTYP(I)) GO TO 550
540   CONTINUE
      PRINT 14,NAME,FILTYP
14    FORMAT(* EMPLIB $$$ FILE TYPE *,A10,* IS NOT ONE OF THE ALLOWED FO
     *RMS *,8A3,*.*)
      JUMP=0
      RETURN
550   IFILE=NTYP(I)
      CALL NEXTWD(CARD,IST,NAME,JFLAG)
      IF(JFLAG) 520,560,540
560   CALL SWAPFIL(IFILE,NAME)
      IF(IFILE.NE.4.AND.IFILE.NE.5) GO TO 510
      IF(IFILE.EQ.4) GO TO 570
      REWIND 5
      IARCH=0
      GO TO 510
570   REWIND 4
      LASTF=0
      JTOC=0
      GO TO 510
600   CONTINUE
C  REFUSE AND SELECT
      IF(JUMP.NE.19.AND.JUMP.NE.20) GO TO 700
      NREC=0
      KFLAG=0
610   CALL NEXTWD(CARD,IST,NAME,JFLAG)
      IF(JFLAG.EQ.2) GO TO 640
      IF(JFLAG) 520,620,630
620   NREC=NREC+1
      IF(NREC.GT.100) GO TO 650
      NAMREC(NREC)=NAME
      KFLAG=0
      GO TO 610
630   KFLAG=1
      GO TO 610
640   IF(KFLAG.EQ.0) RETURN
      READ 15,CARD
15    FORMAT(79A1)
      PRINT 16,CARD
16    FORMAT(1X,79A1)
      KFLAG=0
      IST=1
      GO TO 610
650   PRINT 17
```

```
 17    FORMAT(* EMPLIB $$$ MORE THAN 100 RECORD NAMES GIVEN -- TOO MANY.*
      *)
       JUMP=0
       RETURN
 700   CONTINUE
       IF(JUMP.NE.22.AND.JUMP.NE.23) GO TO 900
 C  REWIND AND ENDFILE
       NAME=IB
       CALL NEXTWD(CARD,IST,NAME,JFLAG)
       IF(JFLAG.EQ.0) GO TO 710
       PRINT 18,NAME,CARD
 18    FORMAT(* EMPLIB $$$ ILLEGAL FILE TYPE.'*,A10,*'*,79A1)
       JUMP=0
       RETURN
 710   DO 720 I=1,8
       IF(NAME.EQ.FILTYP(I)) GO TO 730
 720   CONTINUE
       PRINT 14,NAME,FILTYP
       JUMP=0
       RETURN
 730   J=I
       I=NTYP(I)
       IF(JUMP.EQ.23) GO TO 750
       IF(I.NE.6.AND.I.NE.7.AND.I.NE.4) GO TO 740
 735   PRINT 19,CHAR(JUMP),FILTYP(J)
 19    FORMAT(* EMPLIB $$$ *,A10,* IS AN ILLEGAL DIRECTIVE FOR THE FILE T
      *YPE *,A2,*.*)
       JUMP=0
       RETURN
 740   REWIND I
       NAME=7HREWOUND
 745   PRINT 20,NAME,FILTYP(J),FILNAM(I)
 20    FORMAT(* EMPLIB $$$ *,A9,A2,* FILE NAMED *,A7,*.*)
       RETURN
 750   IF(I.NE.1.AND.I.NE.2) GO TO 735
       ENDFILE I
       NAME=8HENDFILED
       GO TO 745
 800   CONTINUE
 C  SKIP AND SKIPB
       CALL NEXTWD(CARD,IST,NAME,JFLAG)
       IF(JFLAG.NE.2) GO TO 810
       IFILE=1
       RETURN
 810   IF(JFLAG.EQ.0) GO TO 820
       PRINT 21,NAME
 21    FORMAT(* EMPLIB $$$ ILLEGAL NUMBER.'*,A10)
       JUMP=0
       RETURN
 820   IFILE=0
       I=SHIFT(MASK,54)
       DO 850 J=1,3
       L=I.AND.NAME
       IF(L.EQ.(I.AND.IB)) RETURN
       DO 830 K=1,10
```

```
                    IF(L.EQ.(I.AND.SHIFT(NUMS(K),66-6*J))) GO TO 840
                830 CONTINUE
                    PRINT 21, NAME
                    JUMP=0
280                 RETURN
                840 IFILE=10*IFILE+K-1
                850 I=SHIFT(I,54)
                    RETURN
                900 CONTINUE
285             C  REPLACE AND REPLACEB
                    NAMOLD=TOC(1,IFILE)
                    IVOLD=TOC(2,IFILE)
                    CALL NEXTWD(CARD,IST,NAME,JFLAG)
                    IF(JFLAG.NE.1) GO TO 215
290                 NAME=IB
                    CALL NEXTWD(CARD,IST,NAME,JFLAG)
                    IF(JFLAG)230,920,240
                920 IVERS=IB
                    CALL NEXTWD(CARD,IST,IVERS,JFLAG)
295                 IF(JFLAG.LT.0) GO TO 255
                    GO TO 260
                    END
```

```
 SUBROUTINE  NEXTWD                                       CDC 6600 FTN V3.0-P270 OPT=1  01
                        SUBROUTINE NEXTWD(CARD,IST,NAME,JFLAG)
                        INTEGER CARD(1)
                C       JFLAG=-1 IS ERROR, 0 IS NORMAL RETURN, 1 IS COMMA, 2 IS EMPTY CARD
                        NAME=1H
                        JFLAG=2
                        IF(IST.GT.79) RETURN
                        DO 10 I=IST,79
                        IF(CARD(I).EQ.1H,) GO TO 40
                        IF(CARD(I).EQ.1H.) GO TO 15
                        IF(CARD(I).NE.1H ) GO TO 20
                10      CONTINUE
                15      IST=80
                        RETURN
                20      I=I-1
                        JFLAG=0
                        DO 30 J=1,11
                        IST=J+I
                        IF(IST.GT.79) RETURN
                        IF(CARD(IST).EQ.1H,) RETURN
                        IF(CARD(IST).EQ.1H ) RETURN
                        IF(J.LT.11) CALL APPEND(J,CARD(IST),NAME)
                30      CONTINUE
                        JFLAG=-1
                        RETURN
                40      IST=I+1
                        JFLAG=1
                        RETURN
                        END
                SUBROUTINE APPEND(I,CHAR,X)
                DATA MASK/77B/
                ITEMP=SHIFT(MASK,60-6*I)
                JTEMP=SHIFT(CHAR,6-6*I)
5               JTEMP=ITEMP.AND.JTEMP
                X=X.AND..NOT.ITEMP
                X=X.OR.JTEMP
                RETURN
                END
```

```
      SUBROUTINE CPYREC(IIN,IOUT,NAMREC,NREC)
      COMMON /MXC/MX
      COMMON //LMAX,A(1)
      DIMENSION NAMREC(1)
      INTEGER A
      DATA MASK/77B/,IB/1H /
      N=IABS(NREC)
10    CONTINUE
      BUFFER IN(IIN,1) (A,A(LMAX))
      IF(UNIT(IIN)) 30,20,100
20    PRINT 1
1     FORMAT(50X,*-END OF COPY-*)
      RETURN
30    L=LENGTH(IIN)
      IF(L.EQ.0) GO TO 90
      MX=MAX0(MX,L)
      IX=IB
      DO 40 I=1,7
      M=SHIFT(MASK,60-6*I)
      IF((A(2).AND.M).EQ.0) GO TO 50
      IX=(IX.AND..NOT.M).OR.(A(2).AND.M)
40    CONTINUE
50    DO 80 I=1,N
      IF(IX.EQ.NAMREC(I).AND.NREC.LT.0) GO TO 60
      IF(IX.EQ.NAMREC(I).AND.NREC.GT.0) GO TO 70
      IF(I.EQ.N.AND.NREC.GT.0) GO TO 60
      IF(I.EQ.N) GO TO 70
      GO TO 80
60    BUFFER OUT(IOUT,1) (A,A(L))
      IF(UNIT(IOUT).GE.0) GO TO 110
      PRINT 2,IX
2     FORMAT(45X,A7)
      GO TO 10
70    PRINT 3,IX
3     FORMAT(60X,A7)
      GO TO 10
80    CONTINUE
      GO TO 10
90    CONTINUE
      PRINT 6
6     FORMAT(60X,*EMPTY RECORD ENCOUNTERED*)
      GO TO 10
100   PRINT 4
4     FORMAT(* EMPLIB $$$ READ ERROR IN CPYREC.*)
      CALL ABORT
110   PRINT 5
5     FORMAT(* EMPLIB $$$ WRITE ERROR IN CPYREC.*)
      CALL ABORT
      END
```